SECURE MEETING SCHEDULING WITH AGENTA

Thomas Herlea, Joris Claessens, Danny De Cock,
Bart Preneel, Joos Vandewalle
K.U.Leuven ESAT-COSIC, Kasteelpark Arenberg 10, 3001 Leuven-Heverlee, Belgium
firstname.lastname@esat.kuleuven.ac.be
http://www.esat.kuleuven.ac.be/cosic/

Abstract When people want to schedule a meeting, the agendas of the participants must be compared to find a time suitable for all of them. However, at the same time participants want to keep their agendas private. This paper presents a negotiation protocol which tries to solve this contradiction. The protocol is implemented in the agenTa system using mobile software agents, hereby alleviating communication overhead and allowing disconnected operation.

Keywords: mobile agents, secure distributed computation, meeting scheduling

1. INTRODUCTION

When negotiating meetings, the parties look up, communicate and process information about each other's agendas trying to find a moment when they are all free to attend the meeting. Due to the private nature of a person's schedule, as little as possible should be revealed to any other party during negotiation. In the end the result of the negotiation should be known only to the participants and any other information about a user's agenda should remain secret.

Not only people can take part in the process of negotiating meetings. Meeting scheduling naturally accomodates booking for the resources necessary during the meeting. A schedule can be attached to meeting rooms, projection equipment and other infrastructure elements. New meetings can be scheduled taking into account the availability of these resources at different moments in time.

An easy solution for scheduling a meeting is to broadcast the schedules in clear to all participants and pick one of the suitable times according to a common rule. Since this totally neglects privacy we avoid this solution.

Appeared in Proceedings of IFIP CMS 2001, May 21–22, 2001, Darmstadt, Germany,
Communications and Multimedia Security Issues of the New Century
(R. Steinmetz, J. Dittman and M. Steinebach, eds.),
Kluwer Academic Publishers, pp. 327–338.

A mutually trusted third party could be used for scheduling the meeting if it receives the schedules in clear. This way the privacy is protected with respect to the other participants but not with respect to the third party. We also want to avoid this solution.

The aspect of security that our meeting scheduler is most concerned with is user privacy. As such, we have tried to prevent attacks such as spying on private data and result manipulation.

We have analyzed several existing meeting scheduling applications and found that a common problem which needs to be solved by these systems is managing access to a user's agenda. For example, "Yahoo! Calendar" [8] defines access levels for viewing and modifying an agenda entry, and defines user groups to which these access levels are assigned. This is only necessary because the comparison between schedules must be done by the users themselves. Our approach eliminates the need for managing access levels granted to individuals by designing a negotiation protocol that is not based on users directly accessing each other's agenda.

This paper is organized as follows. Sect. 2 presents the negotiation protocol and its analysis from a complexity and security point of view. Sect. 3 discusses the use of mobile software agents and the implementation of our prototype "agenTa" system. Related work is described in Sect. 4. We conclude in Sect. 5.

2. NEGOTIATION PROTOCOL

2.1. GOALS

When designing the negotiation protocol we tried to achieve the following goals, in order to obtain a simple but secure system: parties should have no direct access to each other's agenda, scheduling should only be performed through negotiation; negotiation should only be performed on a limited time span of the agenda; negotiation should only be performed on a chosen subset of the free times; no party should rely on another party telling it the final result; no information should be revealed about the agendas, other than the particular time the meeting can be scheduled.

Scheduling without direct access to the agenda is desirable because it reduces the potential privacy threats and allows to concentrate the concern for protection in one point: the negotiation protocol. In this way, the user does not need to be concerned with the trustworthiness of each of the other participants.

The restrictions of limited time span and subset of the free times ensure that, if something goes wrong during the negotiation only a part of the schedule is revealed. Moreover, chosing a subset of the free times

Appeared in Proceedings of IFIP CMS 2001, May 21–22, 2001, Darmstadt, Germany,
Communications and Multimedia Security Issues of the New Century
(R. Steinmetz, J. Dittman and M. Steinebach, eds.),
Kluwer Academic Publishers, pp. 327–338.

allows users to express their own preferences with respect to scheduling in addition to the restrictions imposed by previous appointments.

Since the negotiating parties are assumed to be mutually distrustful, one of the design goals for the negotiation protocol is that each party arrives to the final result independently. Relying on another party for learning the final result opens a vulnerability to denial of service or misinformation by that party.

Besides information that can be inferred from the final result ("No meeting scheduled" or "Meeting scheduled at moment x"), no information should be revealed about the agendas. This goal protects the privacy of the participants.

2.2. DATA REPRESENTATION

There exists a representation which reduces the problem of deciding if the meeting can be scheduled at a certain moment to a logical AND operation.

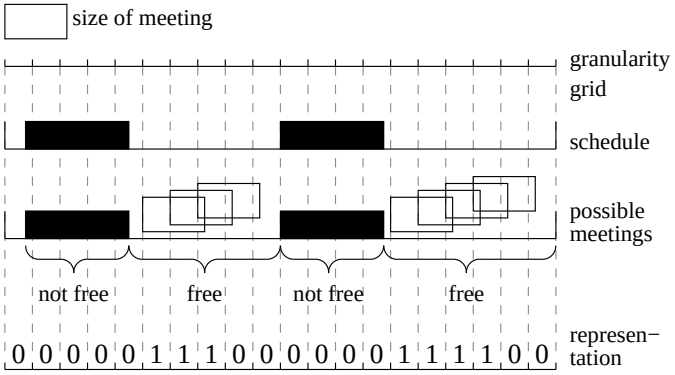

Figure 1 Conversion from agenda to representation

As shown in Fig. 1, an agenda will be represented as a bit string in the following way: for each time slot in the schedule, there is one bit indicating whether the negotiator can (1) or cannot (0) attend a meeting of the specified length which would start at that time. The finer the granularity and the longer the negotiation window, the more bits there will be in the representation.

2.3. SCHEDULING MODEL

In our model, a meeting scheduling starts with an invitation phase. The initiator broadcasts to the invitees a set of negotiation parameters

Appeared in Proceedings of IFIP CMS 2001, May 21–22, 2001, Darmstadt, Germany,
Communications and Multimedia Security Issues of the New Century
(R. Steinmetz, J. Dittman and M. Steinebach, eds.),
Kluwer Academic Publishers, pp. 327–338.

such as meeting length, negotiation window (limited time span in which to attempt the meeting scheduling) and a complete list of invitees. Each invitee broadcasts to all others a reply indicating whether it will accept or decline the negotiation invitation. Because broadcasts are used, no invitee can be mislead as to the set of negotiators it will encounter in the second phase.

In the second phase, called negotiation, the negotiators try the time slots one by one and attempt to schedule the meeting. For each time slot the negotiation takes place according to the protocol outlined below. If the meeting was successfully scheduled the negotiators move on to the third phase, otherwise the next time slot is tried. After independently arriving to a result concerning a certain time slot, each participant broadcasts the result to the others and checks whether all results coincide. This allows for detection of partial failures and attacks which try to mislead a subset of the negotiators.

In the third phase either the common result is presented to the users, or the users are informed that no meeting can take place. If there is a common result, users might confirm their commitment to the scheduled time on a separate channel (e-mail, telephone), independently of the scheduling process.

2.4. SCHEDULING A MEETING

For the purpose of this subsection we will refer to the representation of an agenda as described earlier as "schedule."

Instead of comparing schedules, the negotiation should be based on comparing protected forms of the schedules. The schedules are protected in a way which still allows scheduling to be performed by broadcasting the protected forms to all negotiators and letting them process the data without fear of the unprotected form to be revealed.

The binary XOR operation between the schedule and a mask is a transformation which still allows scheduling to be performed in the sense that the (in)equality of two or more bits is preserved when they are all XORed with the same mask.

If all negotiators know the mask, they are able to retrieve the original schedules easily, by unmasking the broadcasted data. The solution is to let the mask be a shared secret, that is, all negotiators will contribute when building it, but it will not be revealed to any of them.

The negotiation protocol then goes as follows:

1 In step one of the negotiation protocol, each negotiator chooses a random mask, and XORs it with its schedule. This random mask is actually a partial mask. The shared secret will be the XOR of

Appeared in Proceedings of IFIP CMS 2001, May 21–22, 2001, Darmstadt, Germany,
Communications and Multimedia Security Issues of the New Century
(R. Steinmetz, J. Dittman and M. Steinebach, eds.),
Kluwer Academic Publishers, pp. 327–338.

all partial masks, and is called global mask. If even one single negotiator keeps its partial mask secret, the others cannot find the global mask solely using their partial masks.

2 In step two of the protocol, all schedules visit all negotiators exactly one time. At each visit they are masked with the partial mask of that particular negotiator. In the end, all original schedules are thus masked with the global mask, without the need for the negotiators to disclose their partial mask. Since the schedule is first masked with its owner's partial mask it remains secret during its visits.

   A negotiator must be unable to identify a protected schedule as representing its own schedule: otherwise performing XOR between the original and the protected schedule reveals the global mask, allowing the negotiator to retrieve all original schedules. Therefore during the trip to all negotiators, the schedule must be forwarded randomly between the negotiators in order to make it impossible to trace. The schedule must have attached a list of negotiators it hasn't visited yet, decremented at each forwarding, in order to prevent multiple maskings with the same partial mask.

3 In step three, all protected schedules are broadcasted. Each negotiator looks independently for a time slot when all protected schedules have the same value. That implies that the original schedules are identical, too, for that time slot but does not provide any clue whether the negotiators are free or busy for that time slot. The clue is provided by each negotiator's schedule for that time slot. If the negotiator is free then, it means all negotiators are free then and the meeting can be scheduled. For time slots when some are busy and some are free, it is not possible to figure out who the busy ones and who the free ones are.

Note that our scheduling protocol does not specify any form of negotiator authentication. This is however needed for linking the protocol messages to their originators. Depending on the meeting application, the desired form of authentication can be added to the protocol.

Figure 2 shows the negotiation protocol as performed by three parties. For easy understanding of the protocol the schedules in the simulation are following the same route and the maskings appear to be performed simultaneously by the three negotiators. In reality the process is asynchronous (some negotiators may be idle while others are masking) and routing is random (in the end some negotiators may have nothing to broadcast while others may broadcast several protected schedules). An-

Appeared in Proceedings of IFIP CMS 2001, May 21–22, 2001, Darmstadt, Germany,
Communications and Multimedia Security Issues of the New Century
(R. Steinmetz, J. Dittman and M. Steinebach, eds.),
Kluwer Academic Publishers, pp. 327–338.

other difference is that in reality only one bit is processed at a time. If the meeting can be scheduled in the corresponding time slot the protocol stops, otherwise the next time slot is processed.

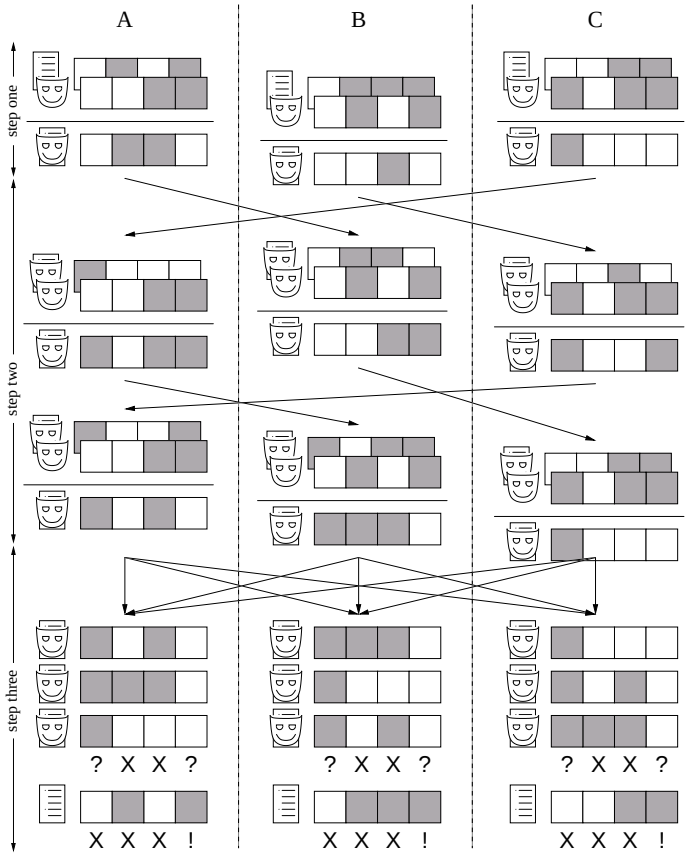

Figure 2 Simulation for three negotiators

2.5. SECURITY ANALYSIS

In this section we list several situations in which the protocol would not completely protect the privacy of the users.

Entropy attack. The reason for performing the negotiation one slot at a time is to prevent the following attack. If the negotiation is done on sequences of slots, when all the broadcasted masked schedules are received, it still is possible for a party to recognize its original schedule. It can be done by testing all the masks which transform the original

Appeared in Proceedings of IFIP CMS 2001, May 21–22, 2001, Darmstadt, Germany,
Communications and Multimedia Security Issues of the New Century
(R. Steinmetz, J. Dittman and M. Steinebach, eds.),
Kluwer Academic Publishers, pp. 327–338.

schedule into one of the protected forms. The correct global mask can be recognized by the fact that by unmasking the other protected schedules with it, bit strings are obtained which have the entropy expected from a schedule.

Negotiating one bit at a time, with fresh partial masks for each bit and stopping when a meeting is scheduled counters this attack because each mask bit and schedule bit have maximal entropy.

Number of parties. When only two parties are negotiating, each can deduce the schedule of the other based on the own schedule and the comparison between the protected forms of the schedules. Besides that, the global mask is straightforward to find because the original schedule can be linked to its protected form.

Three parties are still not enough against this attack on the global mask. When the first negotiator forwards the schedule to the second one, it knows that the protected form will be broadcasted in the end by the third, so it is able to link its original schedule to the protected form and find out the global mask. Since both other schedules can be traced back to their origin negotiators the unmasked schedules can be associated with their users, thereby compromising their privacy.

Also for four negotiators the global mask can sometimes be found out and the originating agents of all unprotected schedules can be deduced. The problem is that the routes the schedules are following are still rather short and allow the schedules to be traced. For five or more participants the ability to trace a schedule along its route decreases as the number of participans increases.

A possible solution is to use dummy negotiators which are always free thereby artificially increasing the number of parties to a level where tracing becomes infeasible. Also, techniques used in anonymous communication are a useful tool in studying and enhancing the untraceability properties of the routing scheme [1].

Bad slots. There may be time slots for which all users are busy and therefore all protected slots will be equal. By checking against the original schedule each negotiator will avoid scheduling a meeting in that slot but it will also know everybody else's schedule for that slot (i.e., everybody is busy). Because they constitute an infringement on all users' privacy we call these slots *bad slots*.

Rogue negotiators. A simple denial of service attack can be mounted by negotiating based on a fully busy schedule instead of declining the invitation.

Appeared in Proceedings of IFIP CMS 2001, May 21–22, 2001, Darmstadt, Germany,
Communications and Multimedia Security Issues of the New Century
(R. Steinmetz, J. Dittman and M. Steinebach, eds.),
Kluwer Academic Publishers, pp. 327–338.

Another possible attack can be performed by a negotiator who randomly changes its partial mask during the negotiation of a time slot. Since the protocol relies on the negotiators consistently using their partial mask for functioning correctly, under this attack the protocol will have unpredictable outcomes.

Goal-oriented misbehavior is also possible. A negotiator can wait to be the last to broadcast the protected schedule(s) it has. This way it is able to detect first when a meeting could take place. In that case it can broadcast a false protected form, preventing the meeting from being scheduled. It knows everybody else's schedule for that time slot, while the others do not.

This attack can be prevented by using time redundancy. If the other negotiators detect the possibility of such an attack they can ask for the round to be reiterated. Therefore, if at each reiteration one and the same negotiator broadcasts results which prevent a meeting to be scheduled, it is almost certain that the negotiator is performing this type of attack. Once identified, the rogue negotiator can be excluded from further steps of the negotiation.

2.6. COMPLEXITY

For analyzing the complexity of the scheduling we count the messages that are sent between the negotiators. In a distributed environment it is expected that sending messages will be much more resource consuming than masking or a comparison between bits. Since much of the processing is done in parallel, bandwidth is more important.

Note that for n negotiators a broadcast is of complexity $n-1$. When an all-to-all broadcast is needed it has complexity $n(n-1)$.

The scheduling starts with a simple broadcast of the invitation. $C_1 = n-1$. All negotiators (except for the initiator) must announce their position towards the invitation. These broadcasts adds complexity $C_2 = (n-1)(n-1)$. For getting masked, one bit must visit all negotiators and then be broadcasted: $2(n-1)$. This happens to each negotiator's bit in a round: $C_3 = 2n(n-1)$. If the number of bits in a schedule is l, after at most l rounds the protocol will end. In the check phase of the scheduling, all negotiators broadcast their result or the fact that no meeting could be scheduled to all others: $C_4 = n(n-1)$. Note that only positive results (i.e., a meeting is possible) are broadcasted. If the result is negative, the agents automatically go to the next bit. If the result is still negative after the last bit, it was not possible to schedule a meeting.

Therefore at most $C = C_1 + C_2 + lC_3 + C_4 = (1+n-1+2nl+n)(n-1) = (2+2l)n(n-1)$ messages are sent. For example, for a scheduling

Appeared in Proceedings of IFIP CMS 2001, May 21–22, 2001, Darmstadt, Germany,
Communications and Multimedia Security Issues of the New Century
(R. Steinmetz, J. Dittman and M. Steinebach, eds.),
Kluwer Academic Publishers, pp. 327–338.

window of 3 eight-hour working days, granularity 1 hour ($l = 24$) and 5 participants ($n = 5$) this amounts to at most 1000 messages; for 10 participants in the same conditions, there will be up to 4500 messages sent.

3. USING MOBILE SOFTWARE AGENTS

3.1. ADVANTAGES

Given the specific conditions of the problem at hand, mobile software agents can bring improvements in at least two areas: network usage and network access (see also [5]).

Due to the large number of messages that have to be sent in the meeting scheduling process, agents alleviate the communication overhead by gathering on an agent host and performing at least part of it locally, saving time and bandwidth.

Agents are a more flexible solution than client-server based solutions. They are loosely coupled entities, therefore their upgrade is less traumatic for the system as a whole.

Concerning network access agents are superior to non-agent approaches because they allow for disconnected operation. The fact that user input is required from users that may not be simultaneously available and potentially long negotiation times stress the superiority of disconnected operation over online operation which keeps underused connections open. Disconnected operation replaces the lengthy connection by two short connections, one when starting the task and one when collecting the result.

3.2. IMPLEMENTATION

We have implemented a prototype of the "agenTa" system described in this paper.

In our prototype implementation we have used the Aglets SDK 1.1 beta 3 [4]. It is a mobile agents system development kit which was recently released to the open source community by its creator, IBM. The SDK contains an agent server, the API needed to write agents in Java, examples and documentation. The prototype implementation of agenTa has around 3500 lines of Java code.

For the inter-agent communication KQML (Knowledge Query and Manipulation Language) was chosen. KQML was developed at the University of Baltimore Maryland County, and enhanced with security capabilities in [7].

Appeared in Proceedings of IFIP CMS 2001, May 21–22, 2001, Darmstadt, Germany,
Communications and Multimedia Security Issues of the New Century
(R. Steinmetz, J. Dittman and M. Steinebach, eds.),
Kluwer Academic Publishers, pp. 327–338.

The agent's feature of mobility is used for performing the negotiation with all agents on the same negotiation host, by local communication. Each user's scheduling application is modular, the user interface, the agenda management and the negotiation being performed by distinct intercommunicating agents.

3.3. EXTENSIONS

Currently, situations like the one in which a participant cannot attend the meeting are dealt with by allowing a participant to decline the invitation. More realistic situations include retreating from the negotiation while it is in progress to avoid blocking the others, relaxation of the schedules such that the chance to schedule a meeting increases or scheduling meetings of teams, in which only several members of a team need to be free in order for the meeting to be scheduled. In order to address these more complex situations, higher level protocols can be devised, which repeatedly use the basic protocol outlined in this paper.

For interoperability and easy interfacing with existing agenda managers, a well defined open interface to agenTa still has to be designed. In essence, an agenda manager which wants to use agenTa to negotiate a meeting in a secure way will only need to send a KQML message with the schedule to agenTa and receive a KQML message with the result from agenTa.

3.4. SECURITY ISSUES

It is generally considered that there are four classes of security issues specific to agents [3]. The first is how to protect the agent host against a malicious agent. The solutions lie in the area of authentication and authorization of the visiting agents. The second issue is protecting agents in transit through the network. This class of problems also has well-established solutions, including VPNs (Virtual Private Networks) between agent hosts. The third issue is protecting the agents against each other. The solutions for this class of problems are more diverse, ranging from relying on the host to ensure proper insulation between agents to using cryptographic techniques for agent authentication and digitally signing messages. The fourth class of problems refers to the protection of an agent against a malicious host it is running on. Since the host executes the agent and has access to its data and code this class is the hardest to solve and we are not aware of any general solutions yet. As other research indicates, the best we can hope for is protecting agents against goal-oriented behavior alterations by the host.

Appeared in Proceedings of IFIP CMS 2001, May 21–22, 2001, Darmstadt, Germany,
Communications and Multimedia Security Issues of the New Century
(R. Steinmetz, J. Dittman and M. Steinebach, eds.),
Kluwer Academic Publishers, pp. 327–338.

Protection against random behavior alteration and denial of service by the host appear to be achievable only with tamper-resistant hardware.

These security issues are typically addressed by the mobile software agent framework. The Aglets framework provides solutions for the first three classes. Agents are isolated against each other and communicate only through messages. Proxy objects are used when agents need references to other agents. Agents travel on the network through encrypted channels. Protection of the host against agents is achieved by authenticating signed incoming agents and establishing appropriate Java sandboxes, as configured in a security policy file. Our "agenTa" system currently does not provide a solution against the fourth class of security problems.

4. RELATED WORK

Alternative approaches and future enhancements of our secure meeting scheduling system can be found in some related work in the area of computer security and cryptography.

The whole purpose of the negotiation protocol is for parties to jointly compute the logical AND of schedule bits coming from each party, without each party having to disclose their own bits. Secure Distributed Computation (SDC) [2] lends itself perfectly to this problem. General secure distributed computation tends to be less feasible than special purpose protocols like the one presented in this paper. However, no comparison has been made in this case.

In step two of the negotiation protocol, a global mask which nobody knows is built from the partial masks of each participant. This is actually an example of a secret sharing mechanism. It is possible that other existing secret sharing mechanisms are suited for the meeting scheduling application.

Step two of the negotiation protocol relies on the random forwarding of data along all the negotiators in such a way that it can not be traced back. This is very similar to techniques used for providing anonymous communication, e.g., on the Internet [1].

The meeting scheduling agents currently have to carry private information when going to a negotiation host. This host has to be trusted as long as the agents cannot be protected from it. The execution of encrypted functions [6] has the potential to protect agents from hosts they run on, and e.g., allow them to digitally sign messages on untrusted platforms.

Appeared in Proceedings of IFIP CMS 2001, May 21–22, 2001, Darmstadt, Germany,
Communications and Multimedia Security Issues of the New Century
(R. Steinmetz, J. Dittman and M. Steinebach, eds.),
Kluwer Academic Publishers, pp. 327–338.

5. CONCLUSION

This article explores the problem of scheduling a meeting without revealing the schedules of the participants. A suitable representation of the schedule is chosen and a protocol for performing the negotiation is presented. The advantages of implementing the protocol with mobile software agents are discussed. Some shortcomings of the protocol are analyzed, solutions and further improvements are suggested.

Acknowledgements

This work was supported in part by the FWO-Vlaanderen project G.0358.99 and by the Concerted Research Action (GOA) Mefisto-666. The second author is funded by a research grant of the Flemish Institute for the Promotion of Industrial Scientific and Technological Research (IWT).

References

[1] J. Claessens, B. Preneel, J. Vandewalle, "Solutions for Anonymous Communication on the Internet," *IEEE ICCST'99*.

[2] R. Cramer, "Introduction to Secure Computation," *Lectures on Data Security - Modern Cryptology in Theory and Practice* Ivan Damgaard, ed., Springer-Verlag LNCS 1561, 1999.

[3] Günter Karjoth, Danny B. Lange, Mitsuru Oshima, "A Security Model for Aglets," *IEEE Internet Computing*, July-August 1997.

[4] Danny B. Lange, Mitsuru Oshima, "Mobile agents with Java: The Aglet API," *World Wide Web Journal*, 1998.

[5] Danny B. Lange, Mitsuru Oshima, "Seven Good Reasons for Mobile Agents," *Communications of the ACM*, Vol. 42, No. 3, 1999.

[6] Tomas Sander, Christian F. Tschudin, "Towards Mobile Cryptography," *IEEE Symposium on Security & Privacy*, 1998.

[7] Chelliah Thirunavukkarasu, Tim Finin, James Mayfield, "Secret Agents - A Security Architecture for the KQML Agent Communication Language."

[8] Yahoo! Calendar, http://calendar.yahoo.com/.

Appeared in Proceedings of IFIP CMS 2001, May 21–22, 2001, Darmstadt, Germany,
Communications and Multimedia Security Issues of the New Century
(R. Steinmetz, J. Dittman and M. Steinebach, eds.),
Kluwer Academic Publishers, pp. 327–338.